Analysis of Algorithms

January 7

What do we look for in an algorithm?

1. Correctness: In the context of this class, this means it gets the exact right answer every time.
2. Speed: Harder to define.

How do we define the speed of an algorithm?

Definition: The \textit{time complexity} of an algorithm is a function $T(n)$ [where $n \in \mathbb{Z}^+$]
where $T(n)$ is the number of steps executed when run on a problem of size $n$, in the worst case.

Note: key decisions made here:

- Worst case: We look at the longest running time for any input of size $n$ because it gives a guarantee of performance.
- Function of input size: Input size impacts the overall run time.
- "Number of steps": We assume each line of pseudocode takes constant time for simplicity/generality.
Example: Consider the following algorithm:

\[ A(\text{list, } x) : \]
1. \text{for } i = 1 \text{ to } n; \\
2. \text{if list}[i] = x, \text{return TRUE} \\
3. \text{return FALSE} \\

Then \( T(n) = n(c_1 + c_2) + c_3 \)

(\text{In the worst case, } x \text{ is not in the list})

We usually make simplifying abstractions to ease the analysis. We drop constant factors and lower order terms from time complexity to get "asymptotic growth rate".

In this case, we say

\[ T(n) = n(c_1 + c_2) + c_3 = O(n) \]

"Big-O" notation:

More formally, we define:

- Definition: A function \( T(n) \) is \( O(g(n)) \)
  
  iff there exists a constant \( c > 0 \) so that
  
  \( T(n) \leq c \cdot g(n) \) for all \( n \) above some threshold

\[ \text{Practically, this is math notation for an upper bound on the asymptotic growth of a function, and for our purposes the function } T(n) \text{ is usually the time complexity of some algorithm} \]
Example: Show \( n(c_1 + c_2) + c_3 \) is \( O(n) \)

Let \( c = c_1 + c_2 + c_3 \)

Then \( cn = (c_1 + c_2)n + c_3 n \)

\[
> (c_1 + c_2)n + c_3 \quad \text{when} \quad n > 1
\]

\[
> O(n)
\]

Example: Show all polynomials \( p(n) = a_0 + a_1n + \ldots + a_d n^d \)

are \( O(n^d) \):

Let \( c = \text{constant} + |a_1| + |a_2| + \ldots + |a_d| \)

Then \( cn^d = |a_0|n^d + \ldots + |a_d|n^d \)

\[
> |a_0| + |a_1|n + \ldots + |a_d|n^d \quad \text{when} \quad n > 1
\]

\[
> a_0 + a_1n + \ldots + a_d n^d
\]

\[
> O(n^d)
\]

Example: Some functions that are \( O(n^2) \):

\( n^2, n^2 + n, 1000n^2 + 1000n, n^{1.999}, n \)

More usefull rules:

- For all \( x > 0 \), \( \log n = O(n^r) \)
- For all \( r > 1, d > 0 \), \( n^d = O(r^n) \)

Polynomials are always preferred over exponentials, regardless of degree or constants.
Related Notation:

Definitions:
- $T(n)$ is $\Omega(g(n))$ if $\exists C > 0$ s.t. $T(n) \geq Cg(n)$ eventually
- $T(n)$ is $\Theta(g(n))$ if $\exists C_1, C_2 > 0$ s.t. $C_1g(n) \leq T(n) \leq C_2g(n)$ eventually
- $T(n)$ is $O(g(n))$ if $T(n)$ is $\Theta(g(n))$ and $\Omega(g(n))$

Example: Show $\sum_{i=1}^{n} i = \Theta(n^2)$:

$O$: $\sum_{i=1}^{n} i \leq \sum_{i=1}^{n} n = n^2 = O(n^2)$

$\Omega$: $\sum_{i=\frac{n}{2}}^{n} i \geq \sum_{i=\frac{n}{2}}^{n} \frac{n}{2} \geq \left(\frac{n}{2}\right)^2 = \Omega(n^2)$

Properties:
- Transitivty: $f = O(g), g = O(h) \Rightarrow f = O(h)$
- Reflexivity: $f(n) = O(f(n))$, same with $\Omega, \Theta$
- Symmetry: $f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$, for $\Theta$ only
- Transpose Symmetry: $f = O(g) \Leftrightarrow g = \Omega(f)$, for $O, \Omega$ only